



(12) **United States Patent**
Hollis

(10) **Patent No.:** **US 11,494,277 B2**
(45) **Date of Patent:** ***Nov. 8, 2022**

(54) **DATA ENCODING USING SPARE CHANNELS IN A MEMORY SYSTEM**

(71) Applicant: **Micron Technology, Inc.**, Boise, ID (US)

(72) Inventor: **Timothy Mowry Hollis**, Meridian, ID (US)

(73) Assignee: **Micron Technology, Inc.**, Boise, ID (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **16/915,413**

(22) Filed: **Jun. 29, 2020**

(65) **Prior Publication Data**
US 2020/0364121 A1 Nov. 19, 2020

Related U.S. Application Data

(63) Continuation of application No. 15/954,149, filed on Apr. 16, 2018, now Pat. No. 10,698,776, which is a (Continued)

(51) **Int. Cl.**
G06F 11/20 (2006.01)
G06F 13/40 (2006.01)
(Continued)

(52) **U.S. Cl.**
CPC **G06F 11/2005** (2013.01); **G06F 11/1008** (2013.01); **G06F 11/221** (2013.01);
(Continued)

(58) **Field of Classification Search**
CPC G06F 11/2005; G06F 11/1008; G06F 11/221; G06F 11/3027; G06F 11/3041;
(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,517,436 A 5/1996 Andreas et al.
5,930,527 A 7/1999 Shin
(Continued)

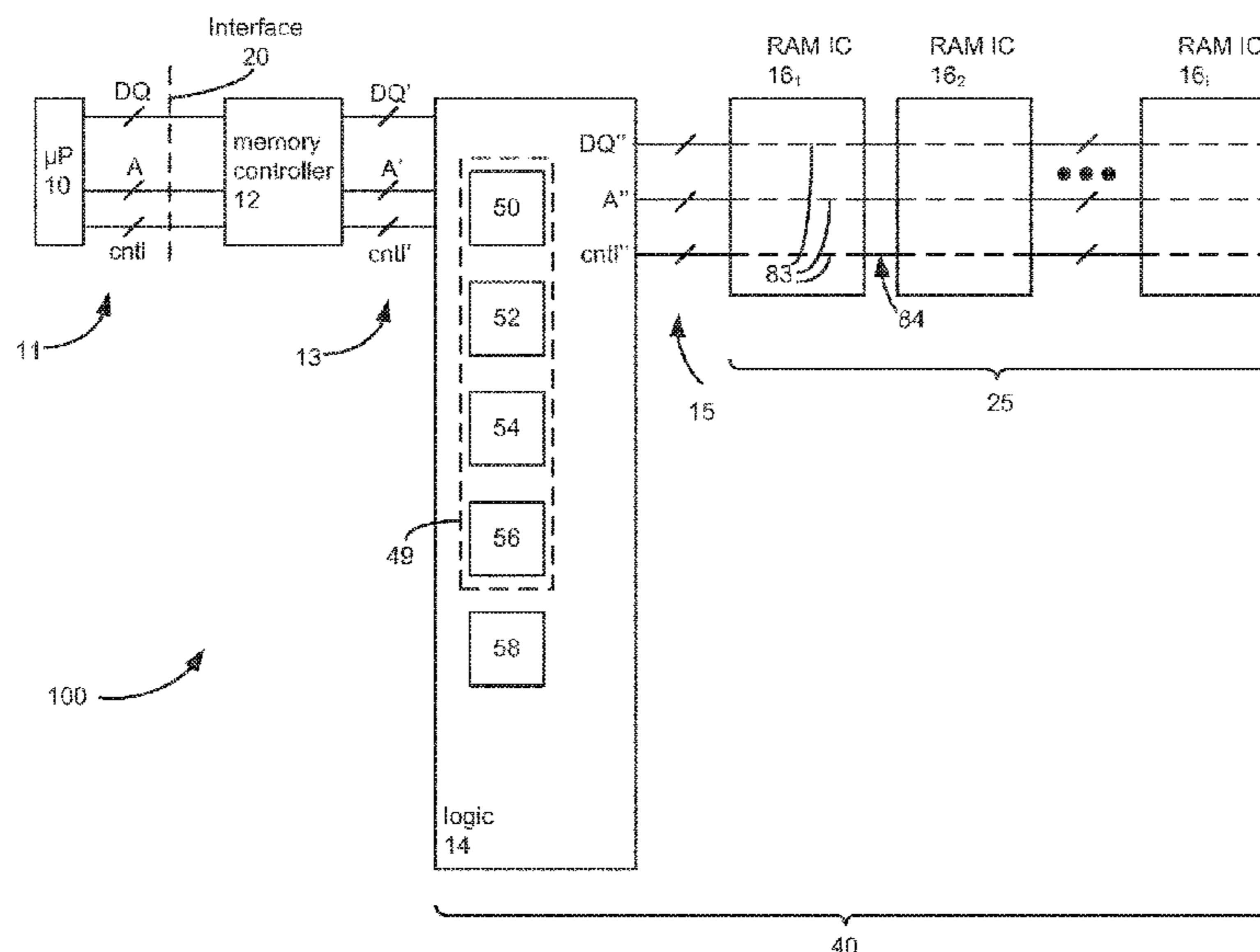
OTHER PUBLICATIONS

"Bus-Invert Coding for Low Power I/O", (1999), 1-20.
(Continued)

Primary Examiner — Henry Tsai
Assistant Examiner — Aurangzeb Hassan
(74) *Attorney, Agent, or Firm* — Schwegman Lundberg & Woessner, P.A.

(57) **ABSTRACT**
Implementations of encoding techniques are disclosed. The encoding technique, such as a Data bus Inversion (DBI) technique, is implementable in a vertically-stacked memory module, but is riot limited thereto. The module can be a plurality of memory integrated circuits which are vertically stacked, and which communicate via a bus formed in one embodiment of channels comprising Through-Wafer Interconnects (TWIs), but again is not limited thereto. One such module includes spare channels that are normally used to reroute a data signal on the bus away from faulty data channels. In one disclosed technique, the status of a spare channel or channels is queried, and if one or more are unused, they can be used to carry a DBI bit, thus allowing at least a portion of the bus to be assessed in accordance with a DBI algorithm. Depending on the location and number of spare channels needed for rerouting, DBI can be apportioned across the bus in various manners. Implementations can also be used with other encoding techniques not comprising DBI.

16 Claims, 17 Drawing Sheets



Related U.S. Application Data

continuation of application No. 14/804,027, filed on Jul. 20, 2015, now Pat. No. 9,946,612, which is a continuation of application No. 12/366,379, filed on Feb. 5, 2009, now Pat. No. 9,087,025.

- (51) **Int. Cl.**
G06F 11/30 (2006.01)
G06F 11/10 (2006.01)
G06F 11/22 (2006.01)
G06F 13/28 (2006.01)
- (52) **U.S. Cl.**
 CPC *G06F 11/3027* (2013.01); *G06F 11/3041* (2013.01); *G06F 13/4022* (2013.01); *G06F 13/28* (2013.01); *G06F 13/4027* (2013.01); *Y02D 10/00* (2018.01)
- (58) **Field of Classification Search**
 CPC .. *G06F 13/4022*; *G06F 13/28*; *G06F 13/4027*; *Y02D 10/00*
 See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,385,739	B1	5/2002	Barton et al.	
6,490,703	B1	12/2002	De la Iglesia et al.	
7,082,489	B2	7/2006	Yeh et al.	
7,154,300	B2	12/2006	Anders et al.	
7,221,292	B2	5/2007	Hein	
7,300,857	B2 *	11/2007	Akram	H01L 21/76877 438/459
7,400,541	B2	7/2008	Jang et al.	
7,549,011	B2	6/2009	Moschopoulos	
7,739,545	B2	6/2010	Check et al.	
7,795,134	B2 *	9/2010	Sulfridge	H01L 27/14683 438/637
9,087,025	B2	7/2015	Hollis et al.	
9,946,612	B2	4/2018	Hollis	
10,698,776	B2	6/2020	Hollis	
2004/0136319	A1	7/2004	Becker et al.	
2008/0017306	A1 *	1/2008	Liu	F16K 99/0036 156/297

2008/0140987	A1	6/2008	Rixner et al.	
2008/0288844	A1	11/2008	Nieuwland	
2009/0019344	A1 *	1/2009	Yoon	G06F 11/1004 714/807
2009/0057745	A1	3/2009	Yin et al.	
2009/0083453	A1	3/2009	Hsueh et al.	
2009/0193159	A1	7/2009	Li et al.	
2009/0193319	A1	7/2009	Shen et al.	
2009/0294954	A1 *	12/2009	Bakir	H01L 25/0657 257/713
2009/0313521	A1	12/2009	Hollis	
2010/0088460	A1 *	4/2010	Jeddeloh	G06F 12/0868 711/103
2010/0118618	A1	5/2010	Kwak	
2010/0199017	A1	8/2010	Hollis	
2015/0324261	A1	11/2015	Hollis	
2018/0232289	A1	8/2018	Hollis	

OTHER PUBLICATIONS

Ghoneima, et al., "Reducing the Data Switching Activity on Serial Link Buses", IEEE Computer Society, Proceedings of the 7th International Symposium on Quality Electronic Design, (Jul. 2006), 6 pgs.

Jang, et al., "Spare Line Borrowing Technique for Distributed Memory Cores in SoC", Instrumentation and Measurement Technology Conference, (May 17, 2005), 6 pgs.

Stan, MR, et al., "Coding a Terminated Bus for Low Power", Great Lakes Symp. on VLSI, (Mar. 1995), 70-73.

Stan, Mircea R, et al., "Bus-Invert Coding for Low Power I/O", IEEE Transactions on VLSI Systems, vol. 3, No. 1, (Mar. 1995), 9 pgs.

Unni, Narayanan, et al., "Enhanced Bus Invert Encodings for Low-Power", IEEE, Circuits and Systems, vol. 5, (2002), 25-28.

Youngsoo, Shin, et al., "Reduction of bus transitions with partial bus-invert coding", Electronic Letters, vol. 34, No. 7, (Apr. 2, 1998), 2 pgs.

U.S. Appl. No. 12/366,379 U.S. Pat. No. 9,087,025, filed Feb. 5, 2009, Data Encoding Using Spare Channels in a Memory System.

U.S. Appl. No. 14/804,027 U.S. Pat. No. 9,946,612, filed Jul. 20, 2015, Data Encoding Using Spare Channels.

U.S. Appl. No. 15/954,149 U.S. Pat. No. 10,698,776, filed Apr. 16, 2018, Data Encoding Using Spare Channels.

* cited by examiner

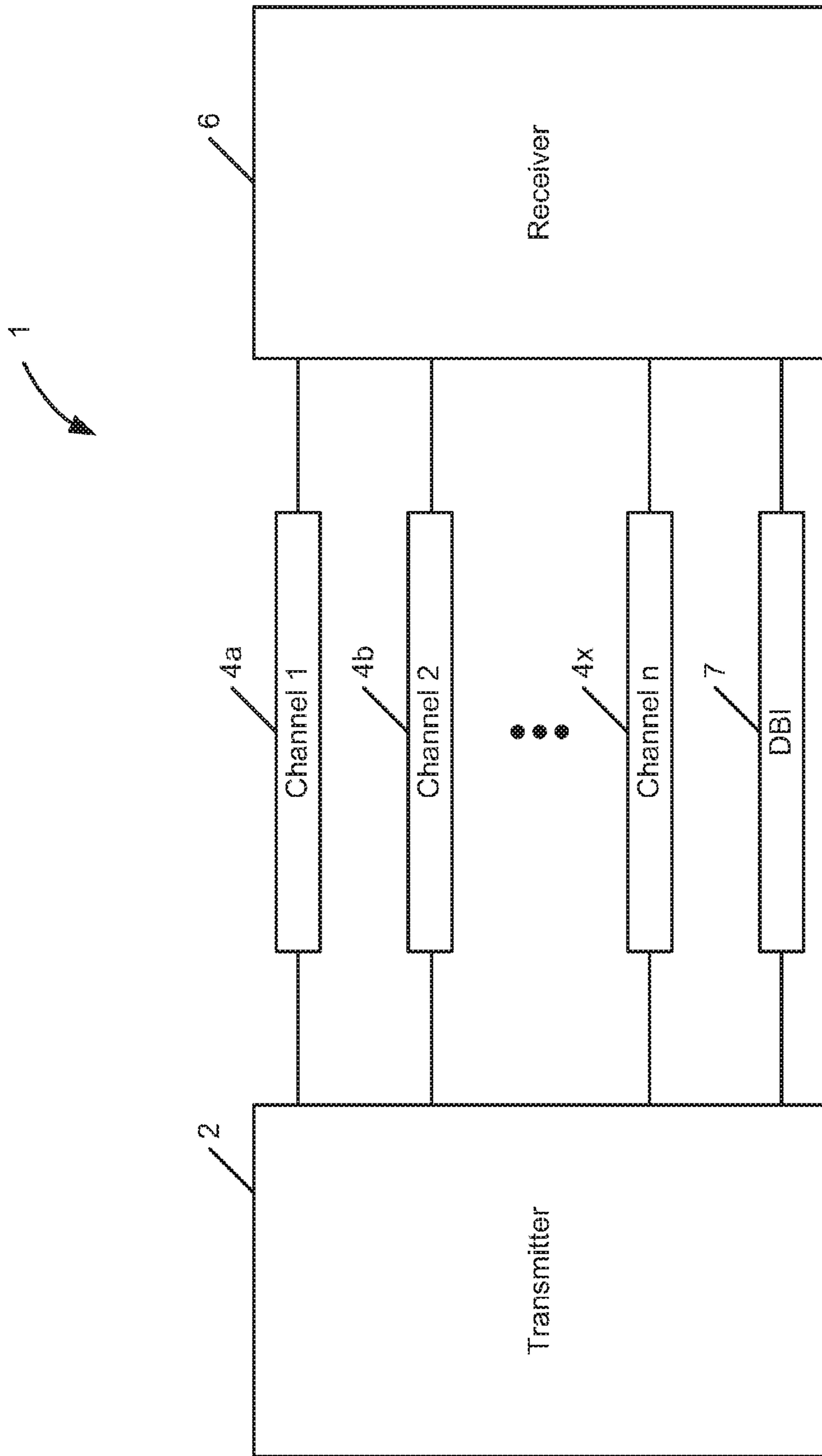


Figure 1
(Prior Art)

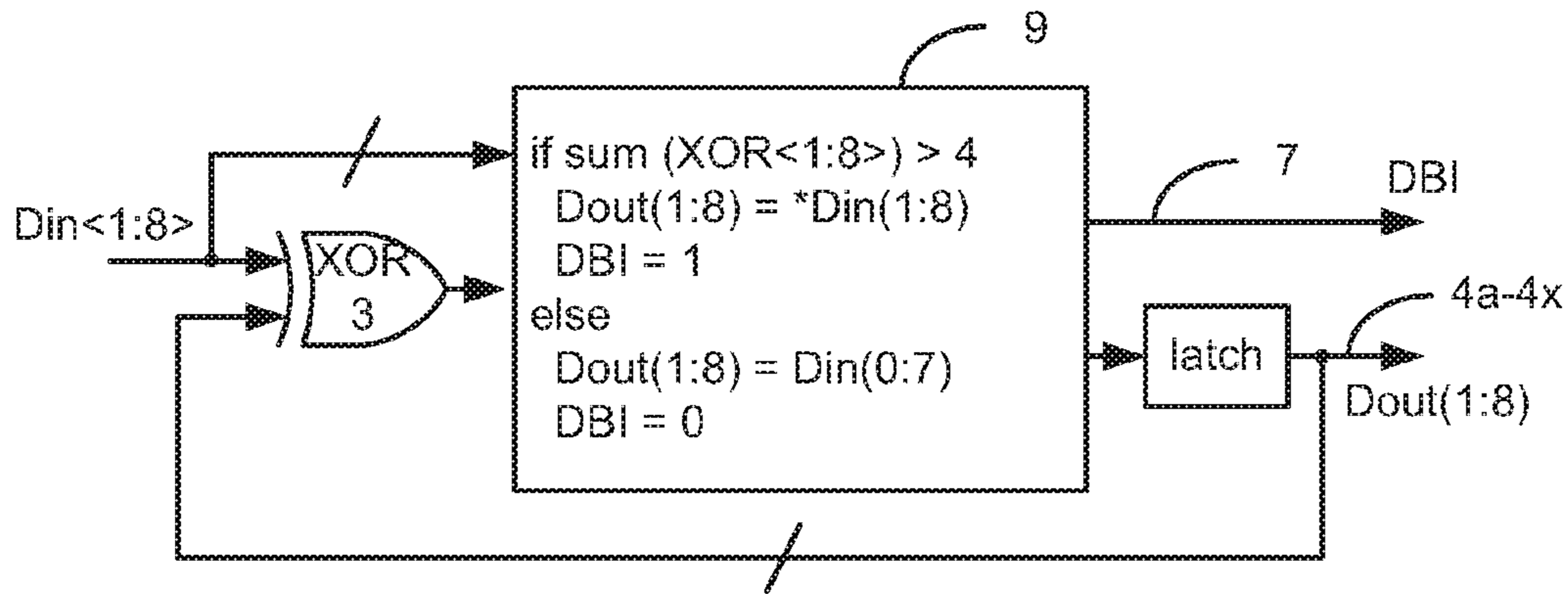


Figure 2A
(Prior Art)

Data without DBI			Data with DBI		
1 2 3 4 5 6 7 8	Transitions		1 2 3 4 5 6 7 8	DBI	Transitions
0 0 0 0 0 0 0 0	N/A		0 0 0 0 0 0 0 0	1	N/A
1 1 1 1 1 1 1 1	8	→	0 0 0 0 0 0 0 0	0	1
0 0 0 0 0 0 0 0	8	→	0 0 0 0 0 0 0 0	1	1
1 1 1 1 1 1 1 1	8	→	0 0 0 0 0 0 0 0	0	1
1 0 1 1 0 1 1 0	3	→	0 1 0 0 1 0 0 1	0	3
0 0 1 1 0 0 1 1	3	→	1 1 0 0 1 1 0 0	0	3
0 1 0 1 0 0 1 0	3	→	1 0 1 0 1 1 0 1	0	3
1 1 0 0 0 1 0 1	6		1 1 0 0 0 1 0 1	1	4
0 0 1 0 1 0 1 0	7	→	1 1 0 1 0 1 0 1	0	2

Figure 2B
(Prior Art)

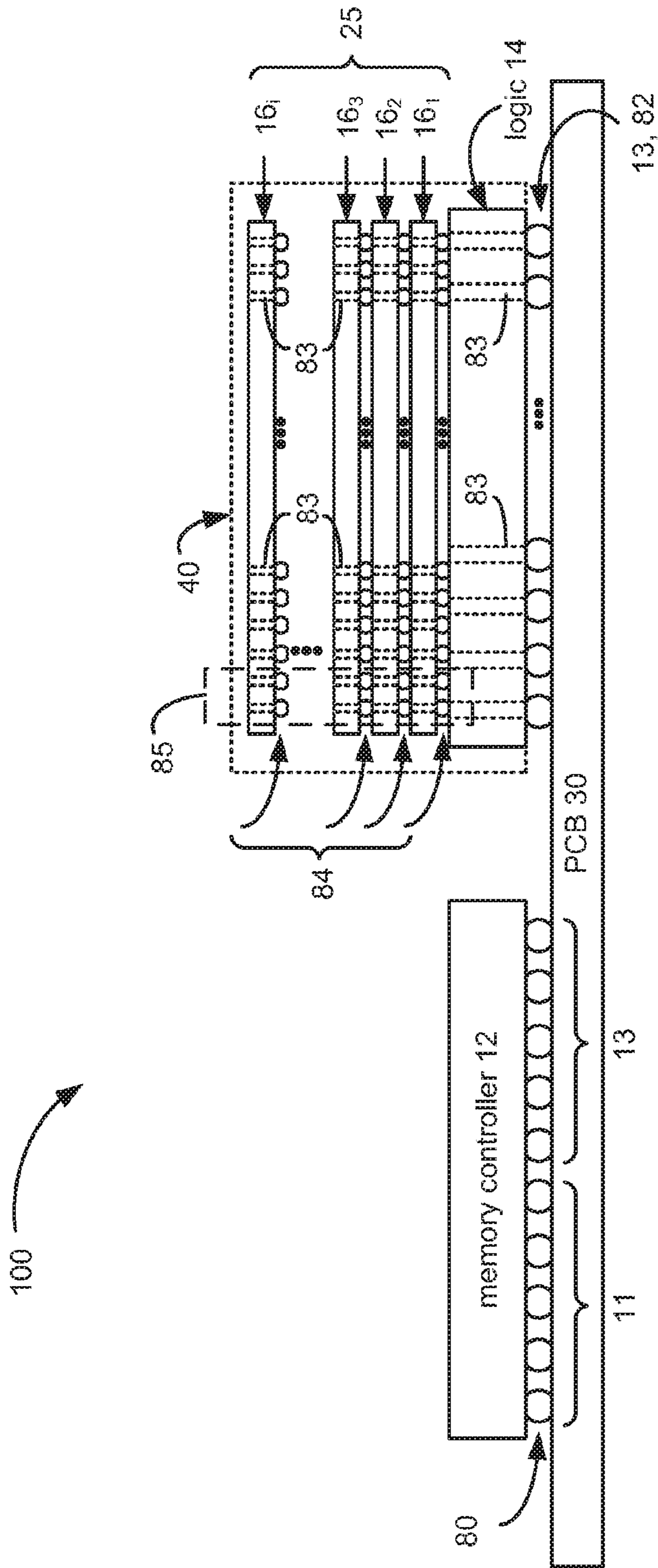


Figure 4

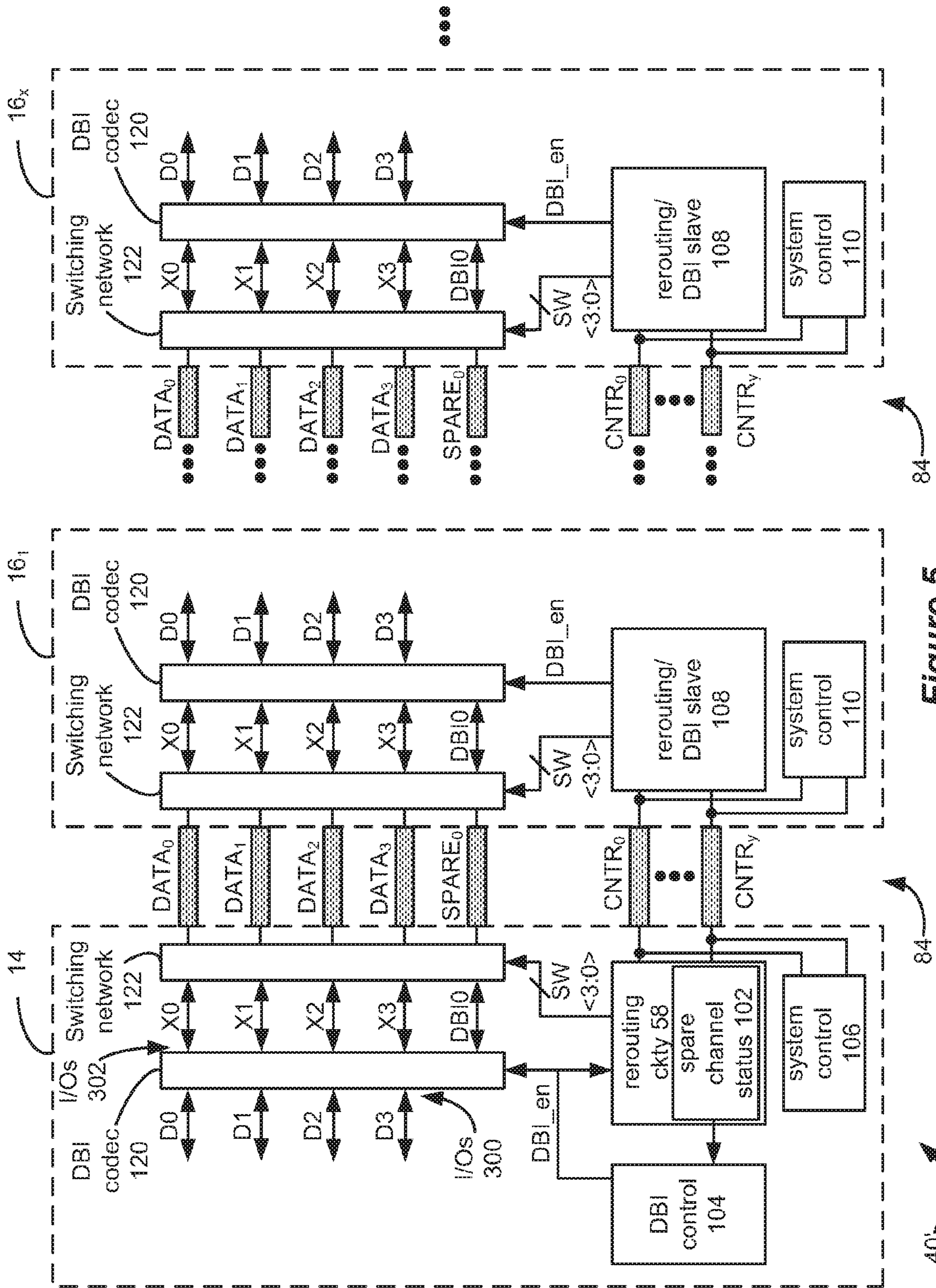


Figure 5

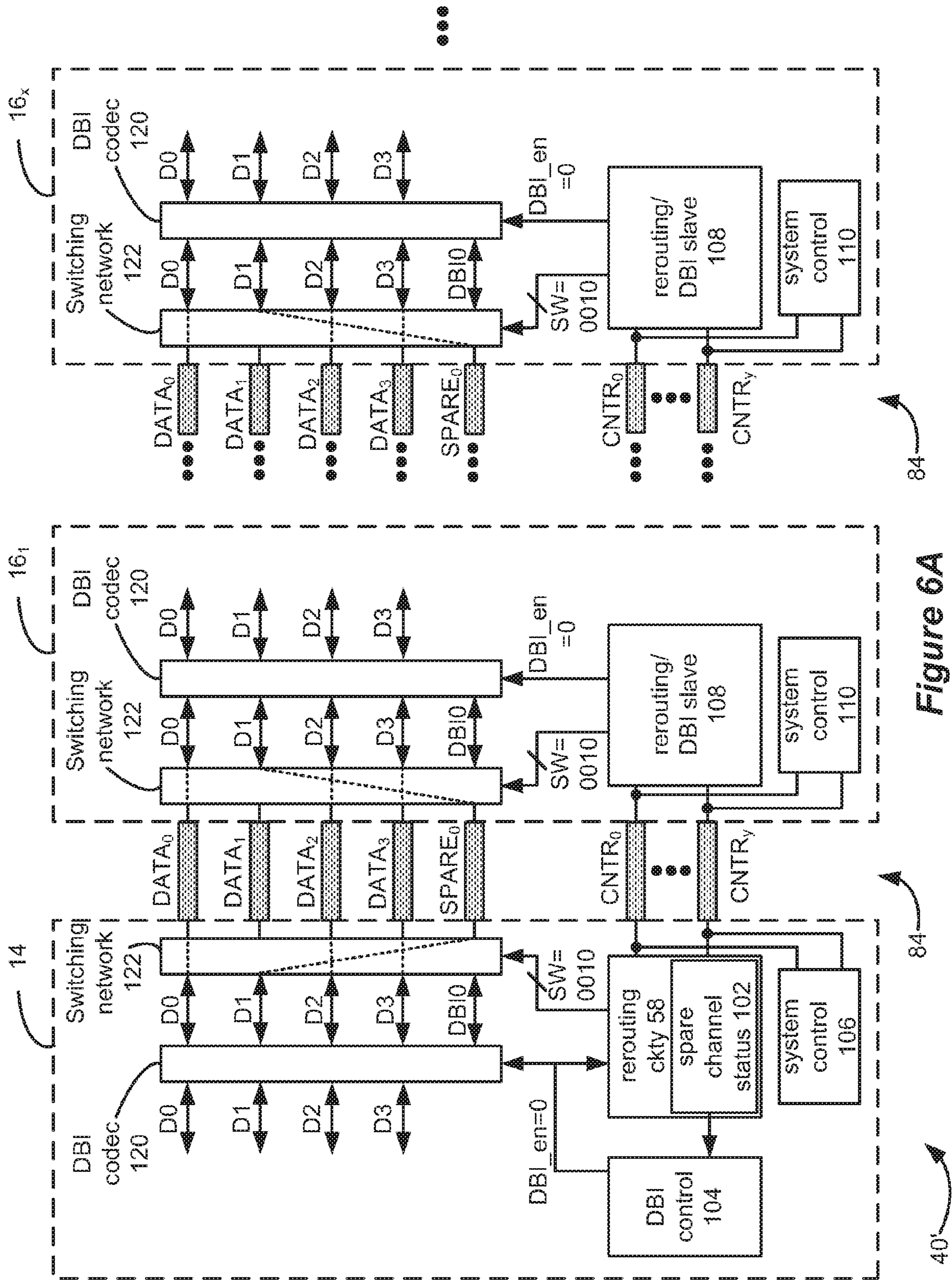
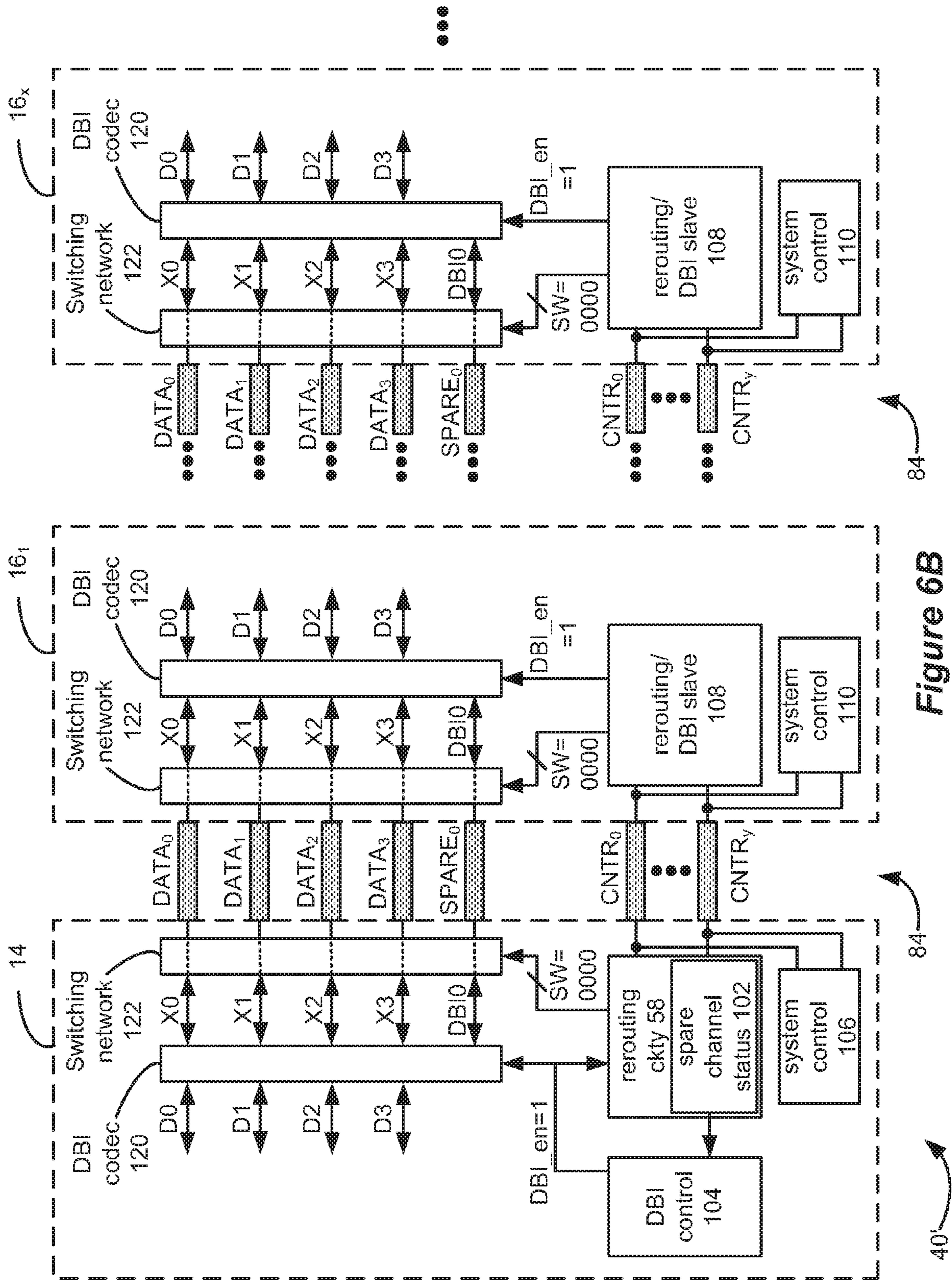


Figure 6A



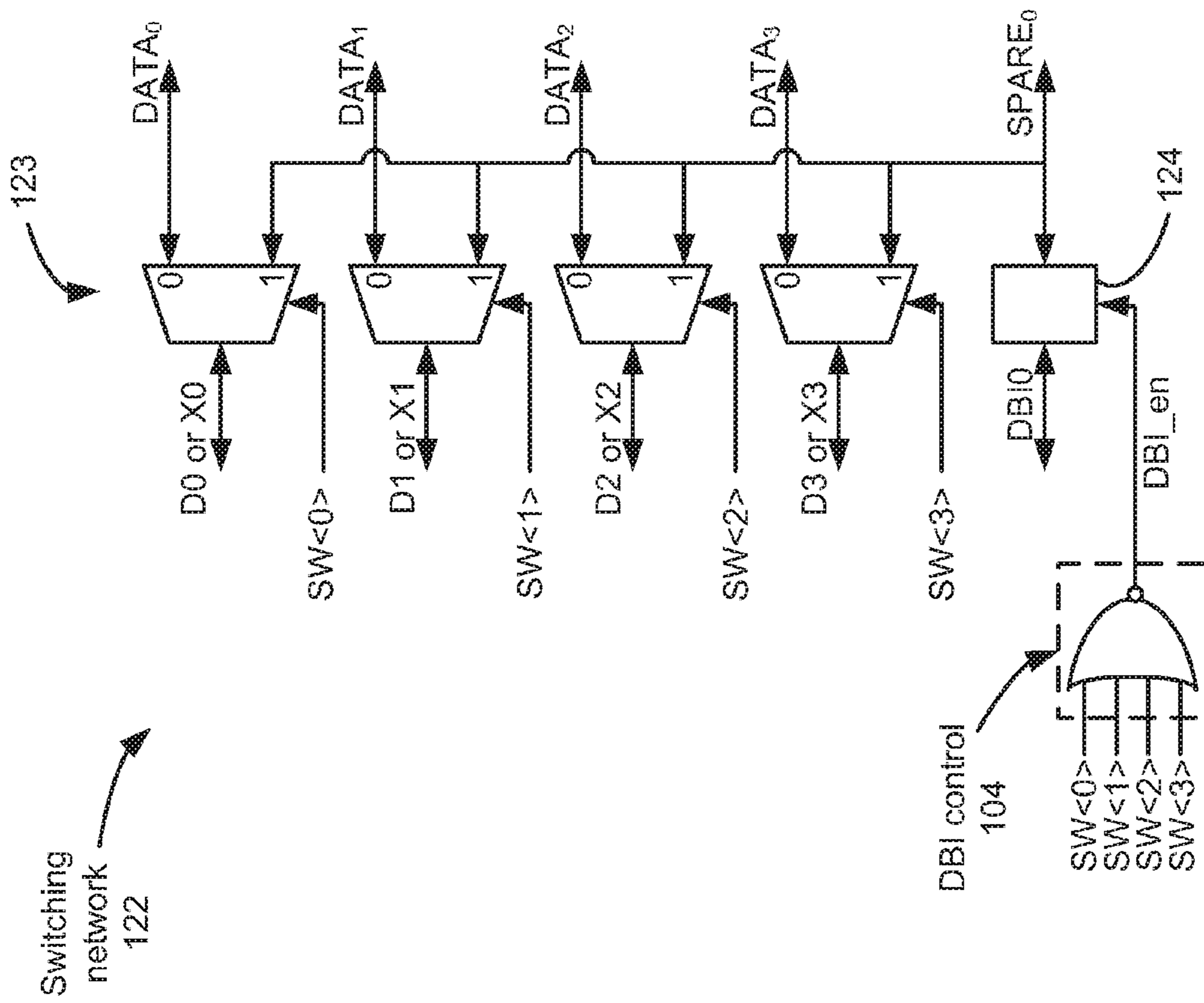


Figure 6C

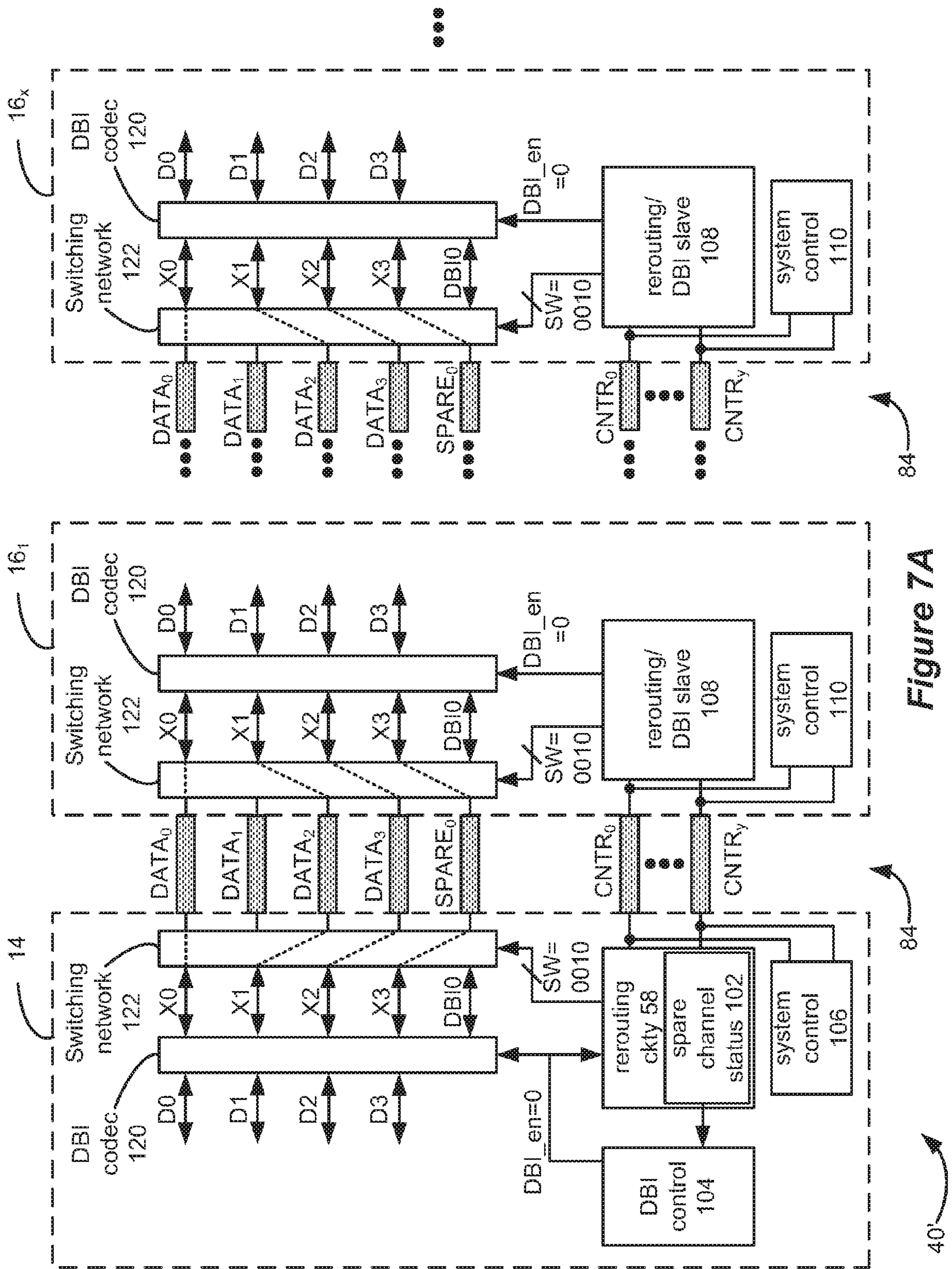


Figure 7A

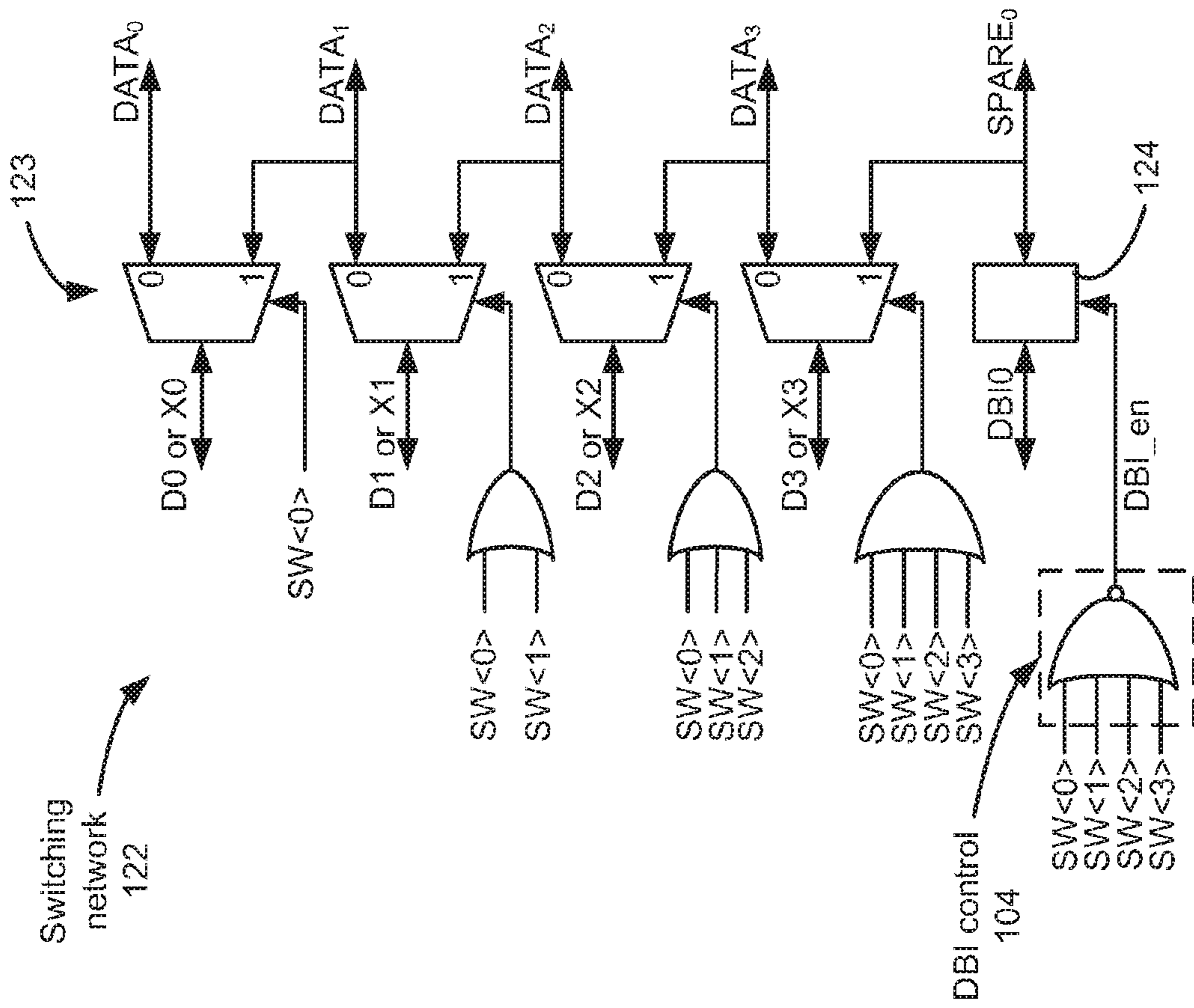


Figure 7B

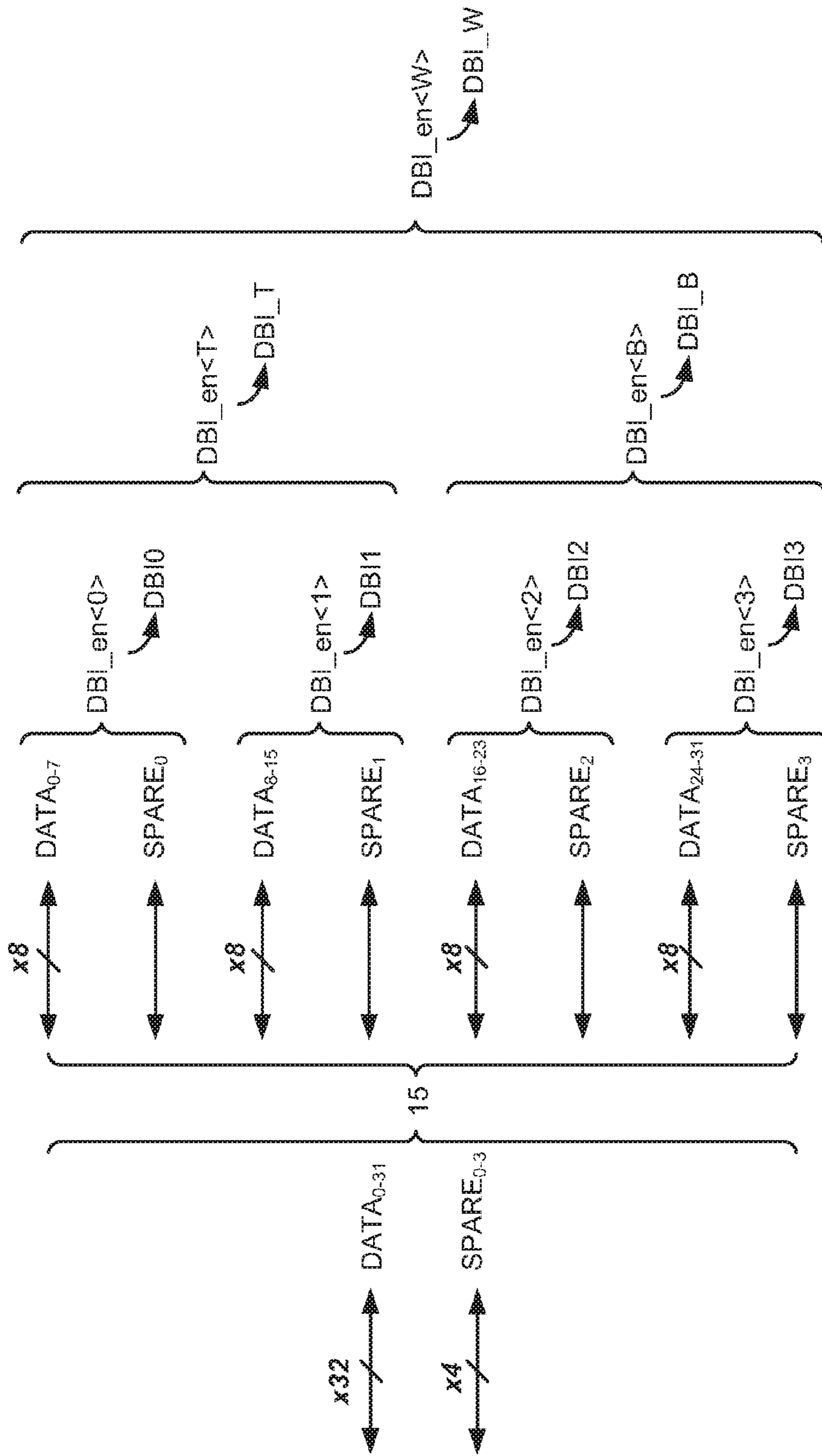


Figure 8

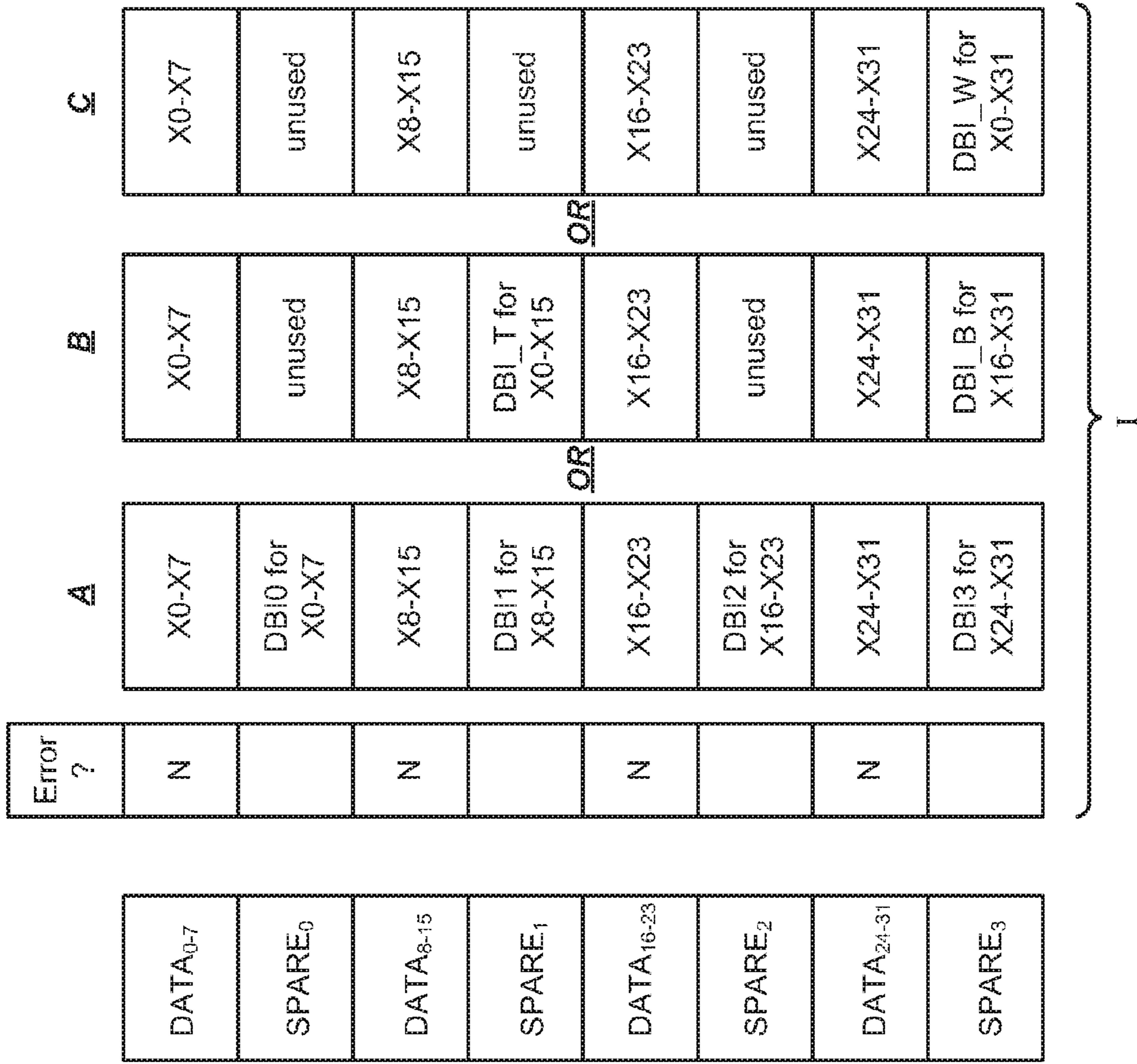


Figure 9A

	III'							
	A		B		C		C	
DATA ₀₋₇	Y	All but affected D0-D7	All but affected X0-X7	All but affected X0-X7	All but affected D0-D7	All but affected X0-X7	All but affected X0-X7	All but affected X0-X7
SPARE ₀		Carries affected D0-D7	Carries affected X0-X7	Carries affected X0-X7	Carries affected D0-D7	Carries affected X0-X7	Carries affected X0-X7	Carries affected X0-X7
DATA ₈₋₁₅	N	X8-X15	X8-X15	X8-X15	All but affected D8-D15	All but affected X8-X15	All but affected X8-X15	All but affected X8-X15
SPARE ₁		DBI1 for X8-X15	DBI_T for X0-X15	DBI_T for X0-X15	Carries affected D8-D15	Carries affected X8-X15	Carries affected X8-X15	Carries affected X8-X15
DATA ₁₆₋₂₃	Y	All but affected D16-D23	All but affected X16-X23	All but affected X16-X23	X16-X23	X16-X23	X16-X23	X16-X23
SPARE ₂		Carries affected D16-D23	Carries affected X16-X23	Carries affected X16-X23	DBI2 for X16-X23	DBI_T for X0-X15	unused	unused
DATA ₂₄₋₃₁	N	X24-X31	X24-X31	X24-X31	X24-X31	X24-X31	X24-X31	X24-X31
SPARE ₃		DBI3 for X24-X31	DBI_B for X16-X31	DBI_W for X0-X31	DBI3 for X24-X31	DBI_B for X16-X31	DBI_W for X0-X31	DBI_W for X0-X31
Error ?								

Figure 9C

DATA ₀₋₇								
SPARE ₀								
DATA ₈₋₁₅								
SPARE ₁								
DATA ₁₆₋₂₃								
SPARE ₂								
DATA ₂₄₋₃₁								
SPARE ₃								

Error ?								
	Y		Y		Y		N	

<u>A</u>		<u>B</u>		<u>C</u>	
All but affected D0-D7	All but affected D0-D7	All but affected D0-D7	All but affected D0-D7	All but affected X0-X7	All but affected X0-X7
Carries affected D0-D7	Carries affected D0-D7	Carries affected D0-D7	Carries affected D0-D7	Carries affected X0-X7	Carries affected X0-X7
All but affected D8-D15	All but affected D8-D15	All but affected D8-D15	All but affected D8-D15	All but affected X8-X15	All but affected X8-X15
Carries affected D8-D15	Carries affected D8-D15	Carries affected D8-D15	Carries affected D8-D15	Carries affected X8-X15	Carries affected X8-X15
<u>OR</u>		<u>OR</u>		<u>OR</u>	
All but affected D16-D23	All but affected X16-X23	All but affected X16-X23	All but affected X16-X23	All but affected X16-X23	All but affected X16-X23
Carries affected D16-D23	Carries affected X16-X23	Carries affected X16-X23	Carries affected X16-X23	Carries affected X16-X23	Carries affected X16-X23
X24-X31	X24-X31	X24-X31	X24-X31	X24-X31	X24-X31
DBI3 for X24-X31	DBI_B for X16-X31	DBI_B for X16-X31	DBI_B for X16-X31	DBI_W for X0-X31	DBI_W for X0-X31

IV

Figure 9D

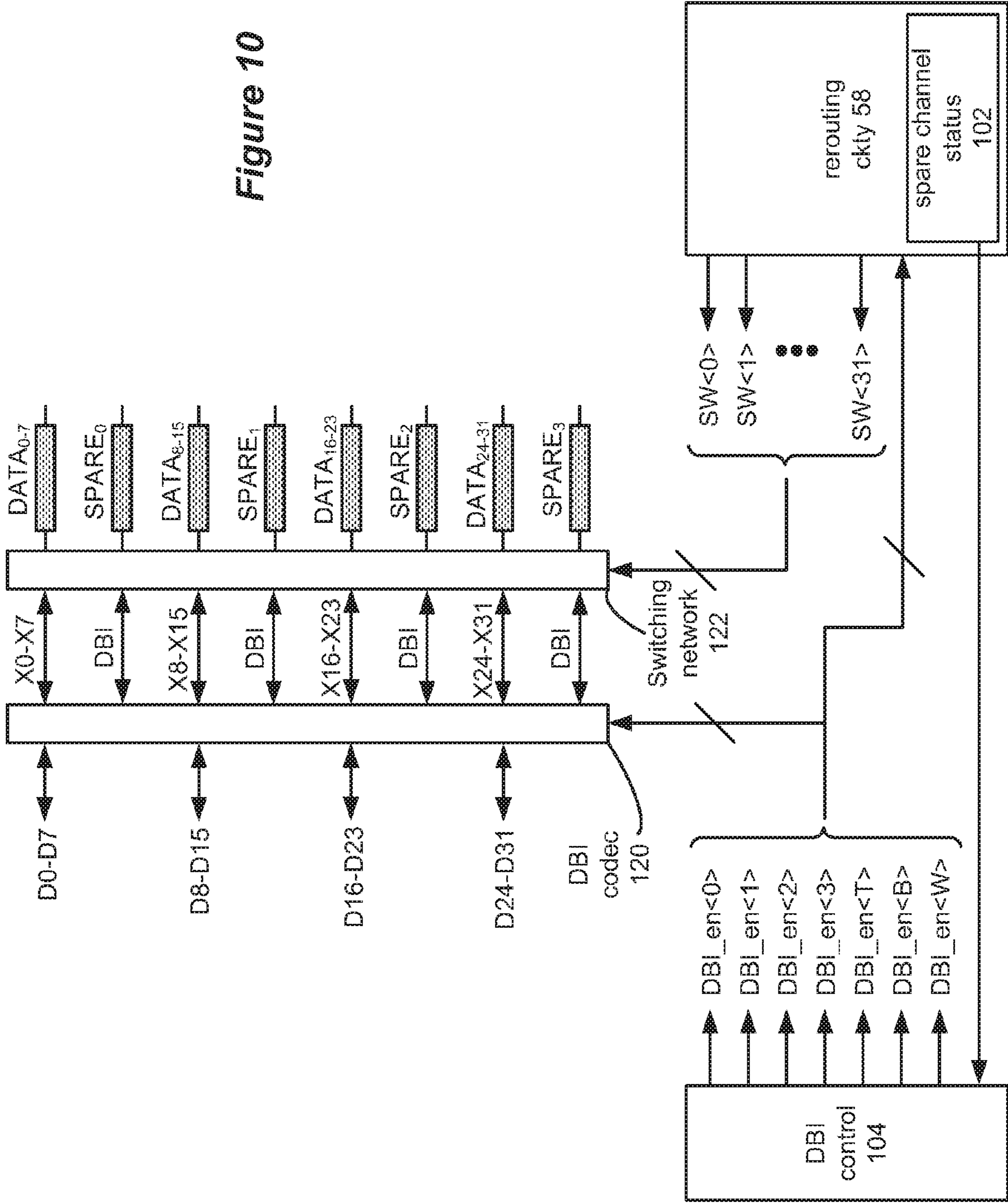
	DATA ₀₋₇								
	SPARE ₀								
	DATA ₈₋₁₅		Y						
	SPARE ₁								
	DATA ₁₆₋₂₃		Y						
	SPARE ₂								
	DATA ₂₄₋₃₁		Y						
	SPARE ₃								
Error ?	Y								
	All but affected D0-D7	Carries affected D0-D7		All but affected D8-D15	Carries affected D8-D15	All but affected D16-D23	Carries affected D16-D23	All but affected D24-D31	Carries affected D24-D31

}

V

Figure 9E

Figure 10



DATA ENCODING USING SPARE CHANNELS IN A MEMORY SYSTEM

This application is a continuation of U.S. application Ser. No. 15/954,149, filed Apr. 16, 2018, now issued as U.S. Pat. No. 10,698,776, which is a continuation of U.S. application Ser. No. 14/804,027, filed Jul. 20, 2015, now issued as U.S. Pat. No. 9,946,612, which is a continuation of U.S. application Ser. No. 12/366,379, filed Feb. 5, 2009, now issued as U.S. Pat. No. 9,087,025, all of which are incorporated herein by reference in their entirety.

FIELD OF THE INVENTION

Embodiments of this invention relate to, among other things, implementation of data bus inversion in a memory system using Through-Wafer Interconnects (TWIs).

BACKGROUND

An example illustrating data transmission between high-speed components within a single semiconductor device, or between two devices in a communication system, is represented by the system **1** shown in FIG. 1. In FIG. 1, a transmitter **2** (e.g., a microprocessor) sends data over one or more transmission channels **4x** (e.g., conductive traces “on-chip” in a semiconductor device or on a printed circuit board) to a receiver **6** (e.g., another microprocessor or memory). As a group, such transmission channels **4x** are often referred to as a “data bus,” which allows one or more data signals to be transmitted from one device to another.

As discussed in U.S. patent application Ser. No. 11/873,779, filed Oct. 17, 2007, a data bus is susceptible to cross talk, simultaneous switching noise, intersymbol interference, and draws power based on the state of the data and/or frequency of data transition. One way to reduce these adverse effects and to prevent unnecessary power consumption is to encode the data. One specific form of data encoding that can be used is Data Bus Inversion (DBI).

Implementation of DBI includes encoding circuitry at the transmitter which assesses the relationship between data bits to be transmitted across a data bus and then decides (based on a particular DBI algorithm) if it would be advantageous to invert some or all of the data bits prior to transmission. If the data bits are inverted, an additional signal, referred to as a DBI bit, is also set at the encoding circuitry to indicate which data bits are inverted. Typically, as shown in FIG. 1, an extra channel **7** is then needed so that the DBI bit may be transmitted in parallel with the data bits to inform the receiving circuitry which groups of data bits have been inverted. The receiver **6** then uses the DBI bit in conjunction with decoding circuitry to return the incoming group of data bits to its original state.

One specific DBI algorithm, illustrated in FIGS. 2A and 2B, is referred to as the “minimum transitions” algorithm. While there may be variations of this technique, in general the minimum transitions algorithm begins by computing how many bits will transition during an upcoming cycle. When more than a certain number of transitions are predicted, encoding circuitry inverts the entire bus, sets the DBI bit to a specified state (high or low depending on the implementation), and drives the inverted data bits and the DBI bit in parallel across the transmission channels **4x** and **7**, with the DBI bit used to decode (i.e., de-invert) the inverted data bits prior to use in the receiver **6**.

The minimum transitions technique can be implemented using the encoding circuitry of FIG. 2A. Because this

technique is discussed at length in the above-mentioned '779 application, it is explained only briefly here. As shown, two successive bytes of data, $Din<0:7>$ (the current byte) and $Dout<0:7>$ (the previous byte), are compared at exclusive OR (XOR) gates **3** on a bit-by-bit basis to determine which bits in the data signals are changing. After this XOR comparison, and in accordance with DBI algorithm **9**, the XOR results are summed, and a determination is made as to whether the sum is greater than four (i.e., whether there are at least five transitions from the previous byte of data to the current byte). If the sum is greater than four, the current byte is inverted before it is transmitted, and the DBI bit **7** is transmitted as ‘1.’ Alternatively, if the sum is four or less (i.e., there are no more than four transitions from one byte to the next), the data is transmitted unaltered, and the DBI bit **7** is transmitted as ‘0.’

FIG. 2B shows how the minimum transitions DBI algorithm **9** works to reduce the number of transitions in a random sequence of bytes, such that no more than four transitions are permitted between successive bytes. Again, this reduction of transitions reduces dynamic current draw at the transmitter **2** and improves the reliability of data transfer by reducing cross talk and simultaneous switching noise.

Different DBI algorithms are beneficial in different circumstances, and not all DBI algorithms are directed to minimizing the number of data transition across transmission channels. For example, other well-known DBI algorithms include the “minimum zeros” algorithm and the “minimum ones” algorithm. The purpose of these algorithms is, respectively, to minimize the number of binary zeros or binary ones transmitted across a channel. Such minimum zeros or ones algorithms conserve power when the driver or receiver circuits coupled to the transmission channels are referenced to VDDQ and VSSQ, respectively, through a resistive termination, and therefore will draw more power when transmitting or receiving a particular data state. For example, if a pull-up resistor connected to the voltage supply is used in a particular driver circuit, driving a logic ‘0’ will require more power than would driving a logic ‘1’. As a result, use of a minimum zeros DBI algorithm would be warranted. Likewise, if a pull-down resistor is used, a minimum ones algorithm would be warranted. The minimum transitions and either of the minimum zeros or ones algorithms can also be combined in a DBI algorithm, as is disclosed in U.S. patent application Ser. No. 12/015,311, filed Jan. 16, 2008. In another DBI algorithm, discussed in the above-mentioned Ser. No. 11/873,779 application, only a portion of the data bits on a bus are inverted to balance the logic states in an encoded byte across the bus, which can be referred to as a Balanced DBI algorithm. Regardless of the DBI algorithm used, all of these DBI algorithms have the common feature of sequentially receiving groups of N original data signals and selectively encoding each group to form a corresponding group of N encoded data signals while issuing at least one encoding (DBI) indicator associated with each group of the N encoded data signals.

A system **100**, potentially benefiting from the implementation of DBI is shown in FIGS. 3 and 4, which is disclosed in U.S. patent application Ser. No. 12/136,868, filed Jun. 11, 2008. System **100** comprises a system for reading/writing from/to a memory set **25**, which in the illustrated example comprises RAM ICs **16_x**. In such a system, it can be assumed for illustrative purposes that, a microprocessor **10** reads from and writes to the memory set **25**, but this is merely one example and any other device or system could be used to so query the memory. In one embodiment, the RAM ICs **16_x** in the memory set **25** comprise dynamic RAM (DRAM) arrays

of memory cells, but could also comprise static RAM (SRAM) cells, or various other types of write/erasable non-volatile memory (e.g., NAND Flash cells, Flash EPROM cells, etc.).

Loom Intervening between the microprocessor **10** and the memory set **25** is a memory controller **12**. Memory controllers **12** are well known in the art and work to create a standard interface **20** with which the microprocessor **10** can predictably communicate. The memory controller **12** couples to the microprocessor's data (DQ), address (A), and control (cntl) busses **11**, and converts them to new busses **13** DQ', A', and cntl' suitable for interfacing with a logic integrated circuit (IC) **14**, discussed further below. Memory controller **12** typically comprises an integrated circuit separate and independent from other components in the system **100**, but this is not strictly necessary, and the controller **12** could be integrated with other components if desired. A high speed differential interface between the controller **12** and the logic chip **14** may exist to improve the overall bandwidth of the system.

In the disclosed embodiment, a logic IC **14** intervenes between the memory controller **12** and the RAM ICs **16_x**. The logic IC **14** contains much if not all of the logic circuitry **49** typically present on a standard RAM IC. For example, the logic IC **14** can contain command decode and queuing circuitry **50**. Such circuitry **50** interprets the various command signals on the cntl' data bus (such as signals write enable (WE), row address strobe (RAS), column address strobe (CAS), and chip select (CS), assuming the RAM arrays **16**, comprise DRAM memory), and issues and organizes the commands as appropriate for distribution to the RAM ICs **16_x** along a control bus cntl". The logic IC **14** may also contain redundancy circuitry **52** for determining faulty memory addresses in the modified RAM ICs **16_x** and for rerouting around such defective addresses to functioning memory cells using programmable fuses or antifuses, as is well known. Logic IC **14** may additionally contain error correction circuitry **54**, which can comprise well known circuitry for assessing and correcting faulty data in accordance with any number of error correction algorithms. Further, logic IC **14** may contain test mode circuitry **56**, which is typically used during manufacturing and/or under the application of special test commands to test the operation of the various RAM ICs **16_x**. Typically, such circuits **50-56** are formed as part of the peripheral logic of a standard memory integrated circuit (not shown), but in the illustrated system such circuitry has been removed from the RAM ICs **16_x**.

System **100** can be manufactured as shown in FIG. **4**, which shows the logic IC **14** and the RAM ICs **16_x** integrated in a multichip module **40** such that they are vertically stacked within a singular package. The bus **15** between the logic IC **14** and the RAM ICs **16_x** is formed using Through-Wafer Interconnects (TWIs) **83**. (If the integrated circuits at issue are silicon-based, these may also be known as Through-Silicon Vias (TSVs)). As is known, TWIs **83** run through at least a portion of the logic IC **14** and through, at least a portion of the, modified RAM ICs **16_x**. The TWIs **83** comprising the bus **15** are connected in series using solder bumps **84**. The module **40** can likewise be affixed to the printed circuit board using solder bumps **82**.

Logic IC **14** may also contain additional integration circuitry relevant to the modular integration of the RAM ICs **16_x**. For example, TWI rerouting circuitry **58** (FIG. **3**) can assess the operation of the various TWIs **83** comprising bus **15**, and if necessary can reroute around any connections deemed to be faulty. To facilitate such rerouting, and as

shown in FIG. **4**, some of the TWIs **83** can comprise spare TWIs **85** which are used in the event that an otherwise prescribed TWI **83** is, or has become, faulty. In this case, switching circuitry on both the logic IC **14** and the RAM ICs **16_x** (not shown) are used to affect the rerouting under the control of signals from the TWI rerouting circuitry **58** as is discussed in further detail later. Circuitry and techniques for rerouting of signals on a TWI-based bus are disclosed in U.S. patent application Ser. No. 12/166,814, filed Jul 2, 2008; Ser. No. 12/173,722, filed Jul. 15, 2008; Ser. No. 12/242,325, filed Sep. 30, 2008; and Ser. No. 11/873,118, filed Oct. 16, 2007.

As recognized in the above-mentioned '868 application, DBI can be used with the TWI-based bus **15**. However, although the use of DBI can reduce power draw in a system such as system **100**, it is generally not desirable to have to provide additional signaling on the bus to provide the DBI channels, such as channel from FIG. **1**. In a system like that depicted in FIG. **4**, the TWI-based bus **15** is already quite complicated, and can comprise hundreds to thousands of TWIs **83**. To add additional TWIs **83** to carry the DBI signals takes up space and adds complication to the design and manufacture of the module **40**.

Nevertheless, the inventor believes that advances in system integration as exemplified by system **100** are making the implementation of DBI, and other data encoding algorithms, more attractive. At the same time, the use of such algorithms is becoming more important as systems shrink and as it becomes increasingly important that such systems reduce their power consumption and operate at high speeds. The inventor believes that it would be desirable to include DBI in a system such as system **100**, without adding additional TWI-based channels to carry the DBI signals.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a block diagram of a data transmission system.

FIGS. **2A** and **2B** illustrates a block diagram and a data table illustrating a minimum transitions data bus inversion (DBI) technique.

FIG. **3** illustrates a system in which RAM ICs are used in conjunction with a logic integrated circuit having circuitry implementing functions traditionally present on RAM ICs.

FIG. **4** illustrates how the system of FIG. **3** can be implemented using a vertically-stacked memory module on a printed circuit board.

FIG. **5** illustrates, in accordance with an embodiment of the invention, improved logic circuitry implementable in the system of FIGS. **3-4** for assessing the status of a spare channel to assist in the application of DBI.

FIG. **6A** illustrates how a data signal designated for a faulty data channel is rerouted to a spare channel using a switching matrix.

FIG. **6B** illustrates that if the spare channel of FIG. **6A** is not used for rerouting, that channel may be used to carry a DBI bit.

FIG. **6C** illustrates the switching network useable to reroute to a spare channel (i) a data signal away from a faulty data channel or (ii) a DBI bit in the event that no data channel is faulty.

FIGS. **7A** and **7B** illustrate an alternative switching network useable to reroute to a spare channel (i) a data signal away from a faulty data channel or (ii) a DBI bit in the event that no data channel is faulty.

FIG. **8** illustrates application of the disclosed technique in the context of a larger bus having a plurality of spare

5

channels, and specifically notes the DBI enable signals that the system may need to consider to apportion the bus from a DBI standpoint.

FIGS. 9A-9E illustrate different manners in which the bus of FIG. 8 can be treated by the DBI control circuitry to implement DBI in different portions of the bus.

FIG. 10 shows circuitry for implementing any of the variations of FIGS. 9A-9E.

DETAILED DESCRIPTION

Implementations of encoding techniques are disclosed. The encoding technique, such as a Data bus inversion (DBI) technique, is implementable in a vertically-stacked memory module, but is not limited thereto. The module can be a plurality of memory integrated circuits which are vertically stacked, and which communicate via a bus formed in one embodiment of channels comprising Through-Wafer Interconnects (TWIs), but again is not limited thereto. One such module includes spare channels that are normally used to reroute a data signal on the bus away from faulty data channels. In one disclosed technique, the status of a spare channel or channels is queried, and if one or more are unused, they can be used to carry a DBI bit, thus allowing at least a portion of the bus to be assessed in accordance with a DBI algorithm. Depending on the location and number of spare channels needed for rerouting, DBI can be apportioned across the bus in various manners. Implementations can also be used with other encoding techniques not comprising DBI.

FIG. 5 shows a general schematic for an improved vertically-stacked memory module 40' generally constructed similarly to the module 40 shown in FIG. 4. As before, the improved module includes a logic IC 14 and a plurality of RAM ICs 16. A parallel bus, akin to bus 15 in FIGS. 3 and 4, carries a plurality of data channels in parallel between the logic IC 14 and the RAM ICs 16_x. Only four data channels (DATA₀-DATA₃) are shown for simplicity. These data channels can carry data, addressing, or control signals, but are assumed to carry the data that is read from or written to the RAM ICs 16_x, for simplicity. Given the vertically-stacked nature of the module 40', the data channels are implemented using TWIs 83 and solder bumps 84, as was depicted in FIG. 4. However, the TWIs 83 are not shown in FIG. 5 for simplicity. As before, the data channels are bidirectional to allow reading and writing from and to the RAM ICs 16_x.

The data channels DATA_i transfer original data signals Di between the logic IC 14 and the RAM ICs 16_x. Generally speaking, and ignoring the distinction between original data signals Di and encoded data signals Xi for the moment, each data signal Di is designated for connection to a particular data channel DATA_i, although this correspondence can vary if rerouting is necessary as explained further below. Absent such rerouting, and assuming a write to the RAM ICs 16_x is occurring, output data path circuitry in the logic IC 14 produces original data signals D0-D3 in parallel, which are then ultimately transferred to data channels DATA₀-DATA₃ respectively, and ultimately to the D0-D3 input data path circuitry in the RAM ICs 16_x. A reading operation occurs similarly, but in the opposite direction.

Also illustrated in FIG. 5 is a spare channel, SPARE₀. SPARE₀ provides an alternative path for a data signal if its designated data channel is (or should become) faulty, for example, because a TWI in that data channel has poor connectivity. Like the data channels, the spare channels can comprise TWIs. Rerouting of the data signals occurs in conjunction with switching networks 122. Jumping ahead to FIG. 6A for a moment, notice via the dotted lines that the

6

switching network 122 operating in the logic IC 14 and each of the RAM ICs 16_x have rerouted a particular data signal D1 away from a faulty data channel (DATA₁) to the spare channel (SPARE₀). Such rerouting occurs under the control of rerouting circuitry 58 operable in the logic IC 14, and rerouting/DBI slave circuitry 108 operable in the RAM ICs 16_x, whose functionality will be explained later.

Returning to FIG. 5, note also the provision of a plurality of control channels (CNTR₀-CNTR_Y). The control channels allow for various control signals to be sent back and forth between the logic IC 14 and the RAM ICs 16_x. Such control signals could comprise the memory-specific control signals discussed earlier (i.e., RAS, CAS, CS, or indicators thereof) or other control signals useful for generally controlling the operation of the module 40'. The nature of the control signals sent on the control channels can generally be dictated by system control circuit blocks 106 and 110 operating in the logic IC 14 and the RAM ICs 16_x respectively. Because general control signals not involved with rerouting are not of particular interest to this disclosure, the details of system control circuit blocks 106 and 110 are omitted.

The control channels CNTR₀-CNTR_Y may be time multiplexed such that they carry different control signals at different points in time. For example, the control channels may carry signals relevant to rerouting only upon initialization of the module 40', or when it is otherwise warranted to make a continuity check of the TWI-based bus of data channels, which is discussed further below. The control channels CNTR₀-CNTR_Y may then carry memory-specific control signals (e.g., from system control circuit blocks 106 or 110) during periods of normal operation, for example.

As just noted, the control channels CNTR₀-CNTR_Y may carry control signals relevant to rerouting around faulty data channels. Such control signals are generated at rerouting circuitry 58 in the logic IC 14, which generally functions during periods of initialization or testing to check for faulty data signals and to perform rerouting to an appropriate spare channel such as SPARE₀. (The above-referenced patent applications discuss such operations further). In the example shown, rerouting circuitry 58 generates four switch control signals SW<3:0>, which are sent to the switching network 122 in the logic IC 14. As will be shown in further detail later, the switching network 122 then reroutes the data signal originally assigned to the faulty data channel onto the spare channel. Indication of these switch control signals SW<3:0> are also sent via the control channels CNTR₀-CNTR_Y to the rerouting/DBI slave circuitry 108 in the RAM ICs 16_x, where they are stored and used to generate matching control signals SW<3:0> to control matching switching networks 122 in the RAM ICs 16_x. The switch control signals can be transported across the control channels CNTR₀-CNTR_Y along one channel, many channels, in serial or parallel, in an encoded fashion, or in any appropriate fashion.

FIG. 6A shows the operation of the switch control signals. In the example shown, rerouting circuitry 58 has determined that data channel DATA₁ is faulty. This causes switch control signal SW<1> in the illustrated example to go high. When this occurs, original data signal D1 is routed to the spare channel SPARE₀ instead to its normally designated DATA₁ signal, as shown by the dotted lines. Switching network circuitry 122 for accomplishing this result is shown in FIG. 6C. Continuing to ignoring the distinction between data signals Xi and Di for the moment, it is seen that the switch control signals SW<3:0> are sent to the control input of multiplexers 123. If a particular data signal Di does not have its corresponding switch control signal asserted (i.e., if SW<i>=0), then that data signal passes to DATA_i. If a

particular data signal (e.g., D1) does have its corresponding switch control signal asserted (e.g., if $SW\langle 1 \rangle = 1$), then that data signal passes to the spare channel, $SPARE_0$. This same routing occurs for data traveling in the opposite direction.

FIGS. 7A and 7B shows alternative switching network circuitry 122. In this embodiment, a faulty data channel is bypassed by sending the affected data signal (e.g., D1) and all subsequent data signals to the next available data channel and to the spare data channel. Thus, as shown by the dotted lines in FIG. 7A, data signal D1 is sent to $DATA_2$, D2 is sent to $DATA_3$, and D3 is sent to $SPARE_0$. Offsetting the affected and subsequent data signals by one channel can be accomplished by using OR logic gates to process the switch control signals $SW\langle 3:0 \rangle$, as shown in FIG. 7B. The operation of such logic gates should be self explanatory to one skilled in the art, and is thus riot further elaborated upon here.

In the event that the spare channel, $SPARE_0$, is not used for rerouting, it may be used for DBI in accordance with embodiments of the disclosed technique, as shown in FIG. 6B. As illustrated, the logic IC 14 and each of the RAM ICs 16_x contains a DBI codec 120 for producing encoded data signals X_i from the original data signals D_i , and for decoding the encoded data signals X_i back into the original data signals D_i . First Input/Outputs (I/Os) 300 couple to the original data signals, while second I/Os 302 couple to the encoded data signals and to the DBI bit (see FIG. 5). So configured, the DBI codec 120 when enabled either encodes the original data signals at the first I/Os 300 into encoded data signals at the second I/Os 302 and issues a corresponding DBI bit at the DBI I/O 302, or decodes encoded data signals at the second I/Os 302 into original data signals at the first I/Os 300 in conjunction with a DBI bit received at the DBI I/O 302. The DBI codec 120 can run any number of DBI algorithms as was discussed in the Background of this disclosure, and no particular DBI algorithm is important to the disclosed technique. DBI encoding and decoding circuitry is well known, and an example can be found in the above-mentioned '779 application.

Regardless of the DBI algorithm employed in DBI codec 120, the algorithm assesses the original data signals D0-D3 and decides whether to invert all or part of those signals. If the DBI algorithm's assessment reveals a benefit to inverting all or a portion of the original data signals D0-D3, then all or a portion of those data signals are inverted to produce encoded data signals X0-X3, and a DBI bit, DBI0, is issued. If the algorithm decides that there is no benefit to inversion, then the original data signals D0-D3 are simply passed as signals X0-X3 without inversion, and the DBI bit is not asserted for that data set.

Whether DBI can be enabled is determined by DBI control circuitry 104. DBI control circuitry 104 essentially determines whether the spare channel, $SPARE_0$, is being used because it is needed for rerouting. If $SPARE_0$ is not being used, DBI control circuitry 104 will enable the DBI codec 120 to apply the DBI algorithm to the original data signals D0-D3 to produce encoded data signals X0-X3 which are forwarded to data channels $DATA_0$ - $DATA_3$, and to provide the DBI bit to $SPARE_0$. If $SPARE_0$ is being used because it is necessary to reroute one of the data signals away from a faulty data channel, DBI control circuitry 104 disables the DBI codec 120 such that the original data signals D0-D3 merely flow through the DBI codec without analysis and without generation of the DBI bit. The switching network 122 then reroutes the data signal affected by the reroute to $SPARE_0$, and the other data signals pass to their respective data channels $DATA_0$ - $DATA_3$ as discussed previously.

To do this, the DBI control circuitry 104 receives an indication of the spare channel status from the rerouting circuitry 58 in the logic IC 14. Specifically, the rerouting circuitry 58 contains spare channel status circuitry 102, which comprises a means for storing an indication of the status of the spare channel. In one simple embodiment, spare channel status circuitry 102 can comprise a four-bit storage register for the switch control signals $SW\langle 3:0 \rangle$, which values are then sent to the DBI control circuitry 104. The DBI control circuitry 104 then analyzes the switch control signals $SW\langle 3:0 \rangle$ to determine whether the spare channel is being used for rerouting or not, and whether the DBI codec 120 can be enabled.

One simple implementation of the DBI control circuitry 104 is shown in FIGS. 6C and 7B, and comprises a NOR gate. Should any of the switch control signals $SW\langle 3:0 \rangle$ be asserted (e.g., $SW\langle 1 \rangle = 1$), indicating a faulty data channel and hence the need for rerouting to the spare channel, the NOR gate outputs a 0 as the DBI enable signal (DBI_en), which disables the DBI codec 120 from applying its algorithm to the data signals D; hence, the original data signals D0-D3 flow through the DBI codec 120 unaffected. Additionally, disabling the DBI enable signal disables pass gate 124, which prevents the DBI bit, DBI0, from passing from the DBI codec 120 to the spare channel. At the same time, the assertion of one of the switch control signal (e.g., $SW\langle 1 \rangle$) causes the affected data signal (i.e., D1) to be rerouted away from its designate data channel (i.e., $DATA_1$), either by routing that data signal to the spare channel, $SPARE_0$ (FIG. 6C), or by shifting the affected and subsequent data signals (FIG. 7B), as discussed previously. Such rerouting is shown in dotted lines in FIGS. 6A and 7A.

Should none of the switch control signals be asserted (i.e., $SW\langle 3:0 \rangle = 0$), meaning that rerouting of data is not necessary, then the NOR gate of DBI control circuitry 104 outputs a 1 as the DBI enable signal, DBI_en. This enables the DBI codec 120 to apply its algorithm to the original data signals D0-D3, to produce encoded data signals X0-X3, and to generate a corresponding DBI bit, DBI0. Additionally, enabling the DBI enable signal enables pass gate 124, which allows the DBI bit, DBI0, to pass to the otherwise unused spare channel, $SPARE_0$. At the same time, the failure to assert any of the switch control signals $SW\langle 3:0 \rangle$ causes the switching network 122 to pass the encoded data bits X0-X3 to their respective data channels $DATA_0$ - $DATA_3$. Such routing is shown in dotted lines in FIG. 6B.

In the technique as described thus far, the logic IC 14, via operation of rerouting circuitry 58 and DBI control 104, comprises the master controllers for rerouting and DBI enablement. The RAM ICs 16_x , by contrast, contain mere corresponding slave controllers, i.e., rerouting/DBI slave circuitry 108. As mentioned earlier, the rerouting/DBI slave circuitry 108 can receive and store rerouting signals from the rerouting circuitry 58 in the logic IC 14 via control channel (s) $CNTR_0$ - $CNTR_y$. The DBI enable signal, DBI_en, can also be transmitted to the RAM ICs 16_x by the control channel(s) $CNTR_0$ - $CNTR_y$, but it may be more convenient instead to merely generate the DBI enable signal at the RAM ICs 16_x from the switch control signals stored at the slave circuitry 108. To summarize, rerouting/DBI slave circuitry 108 essentially mimics the operation of the rerouting circuitry 58 and DBI control circuitry 104 operable in the logic IC 14 so that the DBI codecs 120 and the switching networks 122 in both the logic IC 14 and the RAM ICs 16_x can be controlled similarly.

FIGS. 8-10 show the extension of the disclosed technique to a more realistic 32-bit data bus. As shown in FIG. 8, a

typical TWI-based bus **15** in a memory module **40'** comprises **32** data channels ($DATA_0$ - $DATA_{31}$) and four spare channels ($SPARE_{0-3}$). When so configured, one spare channel is typically designated to serve as the rerouting channel for a given eight bits of the data channels, but this need not always be the case; spares may service more than a single byte. But in the simple case described below, $SPARE_0$ is designated as the rerouting path should any of data channels $DATA_{0-7}$ prove faulty; $SPARE_1$ is similarly designated for $DATA_{8-15}$; etc.

A larger bus with a plurality of spare channels increases the sophistication with which DBI algorithms can operate, assuming that at least one spare channel is not being used for rerouting. For example, and as shown in FIG. **8**, a number of DBI enable signals can be employed to apply DBI to particular portions of the data signals. For example, $DBI_en<0>$ can enable the DBI codec **120** to apply a DBI algorithm to the group of data signals presented to data channels $DATA_{0-7}$, i.e., the first byte of the 32-bit data bus **15**; $DBI_en<1>$ can apply the DBI algorithm to data signals presented to the group of data channels $DATA_{8-15}$; etc. In this way, the bus **15** can be apportioned to provide an independent DBI analysis to four different bytes of the bus. This renders four different DBI bits, DBI_{0-3} : DBI_0 , corresponding to encoding data signals X_0 - X_7 designated for $DATA_{0-7}$; DBI_1 , corresponding to encoded data signals X_8 - X_{15} designated for $DATA_{8-15}$; etc. These DBI bits, DBI_{0-3} , can generally be sent on their respective spare channels $SPARE_{0-3}$, as explained in more detail below.

Apportioning the bus **15** in this manner for DBI purposes adds complexity, but can improve the benefits of DBI. This is because dividing the bus into groups of data signals, and applying DBI to each group independently and generating a DBI bit for each portion independently, reduces power consumption when compared to applying DBI to a larger, undivided bus (which undivided bus would only require one DBI bit). Applying DBI to an apportioned bus is addressed in the following references, which are submitted with the Information Disclosure Statement filed with this disclosure: U. Narayanan, "Enhanced Bus Invert Encodings for Low-Power," IEEE, Circuits and Systems, vol. 5, pgs 25-28 (2002); Y. Shin, "Reduction of bus transitions with partial bus-invert coding," Electronic Letters, vol. 34, no. 7 (Apr. 2, 1998); and M. Stan & W. Burleson, "Bus-Invert Coding for Low Power I/O," pgs. 1-20 (1999).

DBI can also operate to assess portions of the bus **15** larger than a byte. For example, the DBI codec **120** can be enabled to apply DBI to a group comprising a word's worth of data, i.e., 16 bits. This can occur by providing two different DBI enable signals to the DBI codec **120**: $DBI_en<T>$, which enables the DBI codec **120** to apply its algorithm to the top word of data, i.e., to the data signals presented to data channels $DATA_{0-15}$; and DBI_en, which similarly enables the DBI codec **120** to apply DBI to the bottom word of data to be presented to $DATA_{16-31}$. Apportioning the bus in this manner requires two DBI bits, DBI_T and DBI_B , to be carried on the spare channels.

Finally, DBI can also operate on the entire double word of data, i.e., a single group of all 32 data signals. The DBI codec **120** can be enabled to do so via signal $DBI_en<W>$, which assesses DBI based on the whole of the data channels. If DBI is applied to the entire bus, only one DBI bit, DBI_W , would be required, which could be carried on any free spare channel.

The ability to apply DBI at these various levels of bus apportionment improves the flexibility with which DBI can be applied to the data bus **15**. However, since the DBI bit or

bits are sent over the spare channels, the options for applying DBI will be constrained depending on which spare channels are needed for rerouting. The DBI control circuitry **104** therefore, after consideration of the status of the switch control signals to understand where rerouting is occurring along the bus **15**, should issue appropriate DBI enable signals to apportion the bus for DBI purposes without inhibiting rerouting. Various conditions illustrating such options are shown in FIGS. **9A-9E**.

FIG. **9A** illustrates a first condition in which none of the data channels $DATA_{0-31}$ has experienced a continuity error. In this circumstance, all of the data channels $DATA_{0-31}$ carry their designated encoded data signals X_1 - X_{31} , and all of the spare channels $SPARE_{0-3}$ are available to carry a DBI bit, and this can occur in different ways depending on the designer's choice for apportioning the bus. In the option of column A, the bus is apportioned into four bytes, with DBI being applied to each byte. This requires the DBI control circuitry **104** to generate four codec enable signals, $DBI_en<0:3>$. Enabling the codec in this manner causes the codec **120** to generate four DBI bits: DBI_0 associated with encoded data signals X_0 - X_7 , DBI_1 associated with X_8 - X_{15} ; DBI_2 associated with X_{16} - X_{23} ; and DBI_3 associated with X_{24} - X_{31} .

In the option of column B, DBI can be apportioned to the top and bottom words of data. This requires the use of enable signals $DBI_en<T>$ and DBI_en, and the generation of two DBI bits: DBI_T associated with the top word of encoded data signals X_0 - X_{15} ; and DBI_B associated with X_{16} - X_{31} . Here, DBI_T can be sent on spare channel $SPARE_1$ (or $SPARE_0$) and DBI_B can be sent on spare channel $SPARE_3$ (or $SPARE_2$), with the other spare channels remaining unused.

In the option of column C, the bus is not apportioned, and DBI is applied to all 32 data signals, requiring the use of enable signal $DBI_en<W>$ and the generation of only a single DBI bit, DBI_W , which can be sent on any of the spare channels, but which is shown in FIG. **9A** as sent on $SPARE_3$. Because none of the data channels are faulty, the other spare channels go unused.

FIG. **9B** illustrates a second condition in which only one of the data channels has experienced a continuity error and requires rerouting. This error is illustrated as occurring in the first byte of data, i.e., in one of data channels D_{0-7} . In this circumstance, the spare channel associated with the affected byte, $SPARE_0$, will need to carry the affected data signal. In option A, DBI is simply not performed on the faulty byte, i.e., $DBI_en<0>$ is not issued by the DBI control circuitry **104**, and DBI_0 is not generated by the DBI codec **120**. Instead, the original data signals D_0 - D_7 pass to their respective data channels $DATA_{0-7}$, except for the data signal designated for the faulty channel, which is sent to the spare channel $SPARE_0$. In short, the DBI codec **120** is bypassed for the first byte of data. Because the remaining spare channels are unused, they can be used for DBI on a byte basis. Thus, enable signals $DBI_en<1:3>$ are issued, which causes codec **120** to encode the original data signals D_8 - D_{31} into encoded signals X_8 - X_{31} , and which generates DBI bits DBI_1 , DBI_2 , and DBI_3 for each byte. These DBI bits are carried by the free spare channels $SPARE_{1-3}$, respectively.

All other options B, C, and D of FIG. **9B** perform DBI on the entirety of the 32 data signals, but with different apportionments. In option B, DBI is performed on the top word, i.e., the one containing the faulty data channel. More specifically, DBI control circuitry **104**, realizing from the switch control signal that $SPARE_0$ must carry the affected encoded data signal, issues $DBI_en<T>$ to perform DBI on the

11

original data signals D0-D15 to produce encoded data signals X0-X15. The resulting DBI bit, DBI_T, is sent to the other spare channel in the word, SPARE₁. In option B, the bottom two bytes are apportioned on a byte basis, and handled the same as option A, with byte-specific DBI bits, DBI2 and DBI3, being carried by the otherwise unused spare channels, SPARE₂ and SPARE₃.

In option C, DBI is performed on a word basis. This requires issuing enable signals DBI_en<T> and DBI_en, which enables the codec 120 to produced encoded data signals X0-X15 and X16-X31, and their associated DBI bits, DBI_T and DBI_B, respectively. As with option B, the faulty encoded data signal continues to be carried by SPARE₀, with the other spare in the word, SPARE₁, carrying DBI_T. On the bottom word, the DBI bit, DBI_B, is sent to one of the bottom word spare channels (e.g., SPARE₃) and the other spare channel (SPARE₂) is unused. Option C thus differs from option B in that the DBI apportionment is applied to equally-sized portions of the data bus, i.e., two words, instead of one word and two bytes.

In option D, DBI is applied to all 32 bits, similar to what occurred in option C of FIG. 9A, with the only difference being that the affected encoded data signal is rerouted to SPARE₀. As shown in FIG. 9B, the remaining two spare channels SPARE₁ and SPARE₂, i.e., those carrying neither DBI_W nor the affected encoded data signal, remain unused.

FIG. 9C illustrates a third condition in which two of the data channels have experienced continuity errors and require rerouting. This third condition can comprise two different scenarios in which the faulty data channels occur in different words (III) or in the same word (III'). Turning to scenario III first, option A is essentially similar to option A in FIG. 9B, in that DBI is applied on a byte basis but simply not performed on the two faulty bytes. In option B, DBI is performed on all data signals, but on a word basis. Therefore, one spare channel in each word carries the rerouted encoded data signal with the other spare channel carrying the DBI bit (DBI_T or DBI_B) for the word. Option C performs DBI on all 32 bits. One spare channel carries the DBI bit for the double word (DBI_W), two other spare channels carry the rerouted encoded data signals, and the remaining spare channel goes unused.

Scenario III', in which both errors occur in the same word, is essentially the same as Scenario III just discussed. However, in option B, where DBI assessment occurs on a word basis, there is no spare channel in the top byte to carry the DBI bit for the top byte (i.e., DBI_T). Therefore, in this scenario, DBI_T bit is routed to the available spare channel in the bottom word (i.e., SPARE₂). Although this generally violates the preference to keep a particular DBI bit in proximity to its associated data, it requires no particular technical challenge to so reroute the DBI bit.

FIG. 9D illustrates a fourth condition in which three of the data channels have experienced continuity errors requiring rerouting. Option A is similar to option A as discussed in FIGS. 9B and 9C, in that DBI is merely not applied to the bytes having the affected data, leaving only one byte (that byte not requiring rerouting) to benefit from DBI. Option B varies from option A in that DBI is applied to a word of data, instead of a mere byte. Here, DBI is ignored in the top word (the byte having two errors), and the spare channels associated with that word merely carry the rerouted original data signals. DBI is however applied to the bottom word, with the spare in the faulty byte carrying the affected encoded data signal, and the remaining spare carrying the DBI bit for the bottom word (DBI_B). In option C, DBI is assessed for all 32 bits. The single DBI bit from this analysis, DBI_W, is

12

issued in the only byte not experiencing an error, and thus whose spare channel is not already carrying rerouted encoded data.

FIG. 9E illustrates a fifth condition in which all bytes of data are experiencing a continuity error. In this case, all of the spare channels must be used to reroute the data signal from its associated byte. Accordingly, there is no room left to carry any DBI bits. The DBI control circuitry 104 thus issues no DBI enable signals, and the DBI codec 120 is bypassed.

Circuitry for implementing the various conditions illustrated in FIG. 9A-9E is shown generically in FIG. 10. As was discussed earlier, the DBI control circuitry 104 receives an indication of the status of the various spare channels from storage 102, and generates DBI enable signals as appropriate. DBI control circuitry 104 would normally comprise a network of logic gates configured to affect DBI in accordance with the various conditions set forth in FIGS. 9A-9E, and in accordance with the particular option within each condition that the designer finds suitable for a given environment. Given the wide array of potential DBI encoding options just discussed, one skilled in the art will realize that the logic network of DBI control circuitry 104 will be implementation specific, and a routine matter to construct once the designer's options have been finalized.

The switching network 122 will likewise be implementation specific and will depend on the DBI options chosen. When applied to a more complex bus such as that shown in FIG. 10, one skilled in the art will realize that the multiplexers in the switching network 122 may require more complex logic function at their inputs, or may require multiple inputs or multiple levels of multiplexing. Designing such logic would be routine given the details provided in this disclosure once the particular options for DBI encoding have been finalized.

Although conceived of in the context of a vertically stacked memory module, it should be noted that the inventive concepts disclosed herein are not necessarily limited to that environment.

Any of these DBI algorithms mentioned above, or other data encoding algorithms more generally, can be used in the context of the disclosed embodiments of the invention. Additionally, other embodiments of the invention would not necessarily require operation of a data bus inversion algorithm. Instead, the codecs 120 could comprise other types of encoders and decoders (e.g., error detection, or error correction), and the DBI bit could more generally comprise an encoding indicator bit or bits (e.g., an error detection bit, or an error correction bit) consistent with the particular decoder. Thus, previously-existing or future-developed encoding/decoding schemes will also benefit from the disclosed techniques.

While some implementations have been disclosed, it should be understood that the disclosed circuitry can be achieved in many different ways to the same useful ends as described herein. In short, it should be understood that the inventive concepts disclosed herein are capable of many modifications. To the extent such modifications fall within the scope of the appended claims and their equivalents, they are intended to be covered by this patent.

What is claimed is:

1. A method comprising:

controlling a switching assembly of a memory device, the switching assembly coupled to at least first and second memory channels, each memory channel including a respective plurality of data channels for a respective

13

word of data and one or more additional channels including at least one data bit inversion (DBI) channel; and

in response to determining that at least one of the data channels of the first memory channel is faulty, rerouting a data signal of the faulty data channel to a DBI channel, and assigning a DBI channel of the second memory channel to operate as a DBI channel for a double word including the respective words of the first and second memory channels.

2. The method of claim 1, wherein each memory channel comprises 16 respective data channels for the respective word of data.

3. The method of claim 1, wherein the state of the assigned DBI channel operates to identify an inversion state of the 32 data channels of the double word.

4. The method of claim 3, wherein the 32 data channels are coupled to adjacent portions of a memory bus.

5. The method of claim 1, further comprising:
 encoding data of a first word to be transmitted through the first memory channel; and
 encoding data of a second word to be transmitted through the second memory channel.

6. The method of claim 5, wherein the encoding is different for the first word than for the second word.

7. The method of claim 6, wherein the encoding of the first and second words is determined in response to a DBI Inversion algorithm.

8. The method of claim 6, wherein the first and second words are to be transmitted within a common time interval, and wherein the encoding of the second word is determined in response to the data of the first word.

9. A stacked memory device, comprising:
 a logic circuit comprising DBI encoding circuitry, data channel remapping circuitry, and test circuitry to test for continuity of individual data channels interconnecting the logic circuit and multiple stacked memory devices;
 multiple stacked memory devices stacked with the logic circuit, the logic circuit and the multiple stacked memory devices interconnected with one another by through wafer interconnects (TWIs), to communicate multiple data, address, and control signals through respective memory data channels from a bus to one or more of the stacked memory devices;
 wherein the data TWIs include further include additional data channels; and
 wherein the data channel remapping circuitry is operable to map one or more of the additional data channels to function as either,
 a replacement data channel in place of a faulty data channel, or
 a DBI indicator channel carrying a signal identifying whether data transmitted on a first group of data channels is inverted;
 wherein in response to a determination by the test circuitry that an individual data channel is faulty, the logic circuitry performs functions including rerouting a data

14

signal of the faulty data channel to a DBI channel, and assigning a DBI channel of a second memory data channel to operate as a DBI channel for a double word including respective words of first and second memory data channels.

10. The stacked memory device of claim 9, wherein the first group of data channels is configured to carry a word of data.

11. The stacked memory device of claim 9, wherein the first group of data channels includes 16 data channels.

12. A stacked memory device, comprising:
 a logic circuit comprising DBI encoding circuitry and data channel remapping circuitry;
 multiple stacked memory devices stacked with the logic circuit, the logic circuit and the multiple stacked memory devices interconnected with one another by through wafer interconnects (TWIs), to communicate multiple data, address, and control signals through respective channels from a bus to one or more of the stacked memory devices;
 wherein the data TWIs include further include additional data channels; and
 wherein the data channel remapping circuitry is operable to map one or more of the additional data channels to function as either,
 a replacement data channel in place of a faulty data channel, or
 a DBI indicator channel carrying a signal identifying whether data transmitted on a first group of data channels is inverted;
 wherein a determination by the test circuitry that an individual data channel of the first group of data channels is defective, causes the channel mapping circuitry to remap an additional data channel from being a DBI indicator channel for the first group of data channels to a data channel replacing the defective data channel.

13. The stacked memory device of claim 12, wherein the channel mapping circuitry is further operable to remap a DBI indicator channel for a second group of data channels to serve as a DBI indicator channel for both the first and second groups of data channels.

14. The stacked memory device of claim 13, wherein the first and second groups of data channels are adjacent channels along the bus.

15. The stacked memory device of claim 12, wherein remapping an additional data channels from being a DBI indicator channel comprises disabling the defective data channel and re-mapping the defective data channel and subsequent data channels in the first group of data channels.

16. The stacked memory device of claim 12, wherein remapping an additional data channel from being a DBI indicator channel comprises disabling the DBI encoder for that additional data channel.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 11,494,277 B2
APPLICATION NO. : 16/915413
DATED : November 8, 2022
INVENTOR(S) : Timothy Mowry Hollis

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On the Title Page


In item (54), in "Title", in Column 1, Line 2, after "CHANNELS", delete "IN A MEMORY SYSTEM"

In item (57), in "Abstract", in Column 2, Line 4, delete "riot" and insert --not-- therefor

On page 2, in Column 2, under "Other Publications", Line 8, delete "Lovy" and insert --Low-- therefor

In the Specification

In Column 1, Line 2, after "CHANNELS", delete "IN A MEMORY SYSTEM"

Signed and Sealed this
Third Day of January, 2023

Katherine Kelly Vidal
Director of the United States Patent and Trademark Office